Reproducible Experiment Platform

Alexander Baranov, Egor Khairullin, Tatiana Likhomanenko, Alexey Rogozhnikov, Andrey Ustyuzhanin

The 21st International Conference on Computing in High Energy and Nuclear Physics, Okinava, 2015

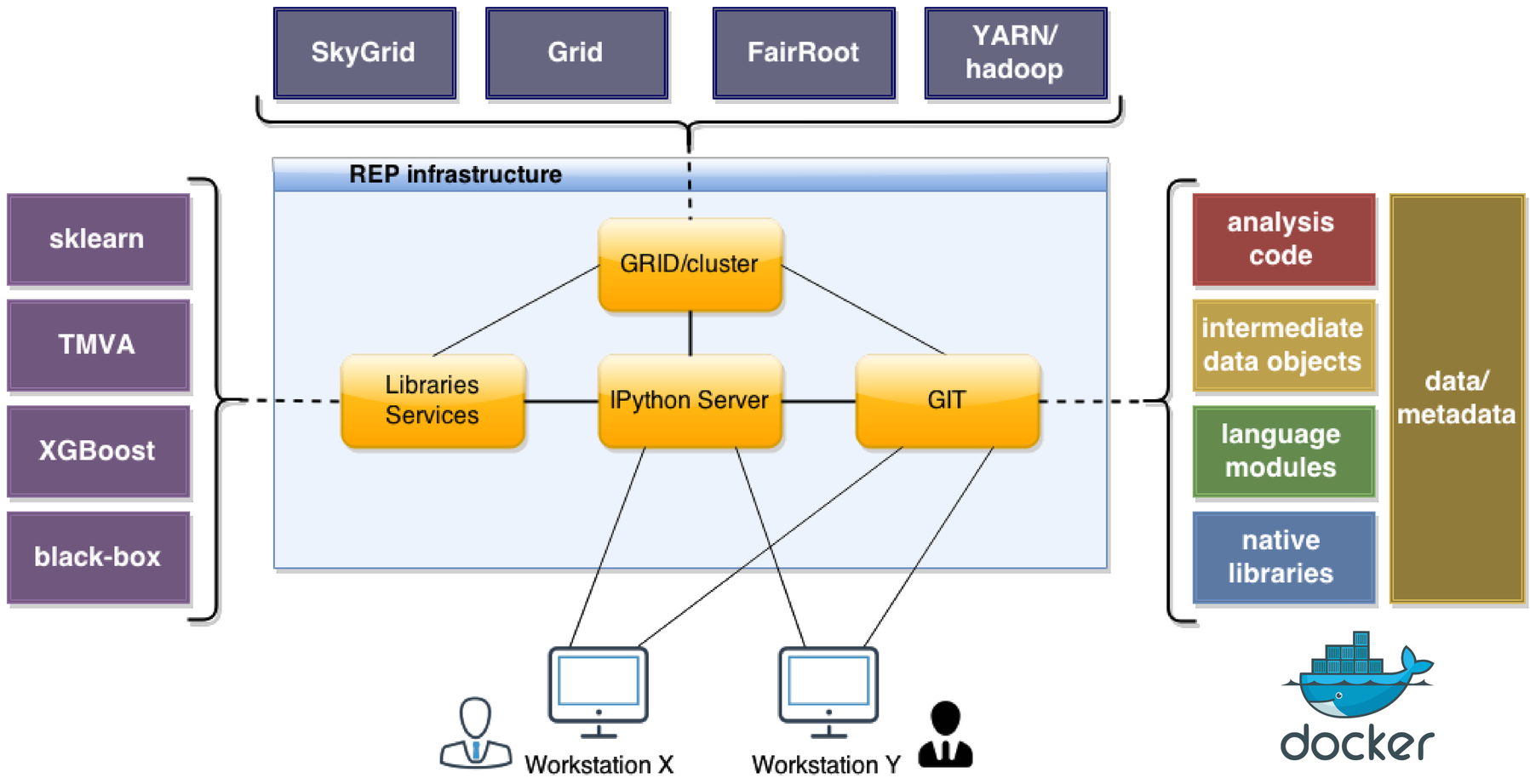
github.com/yandex/REP

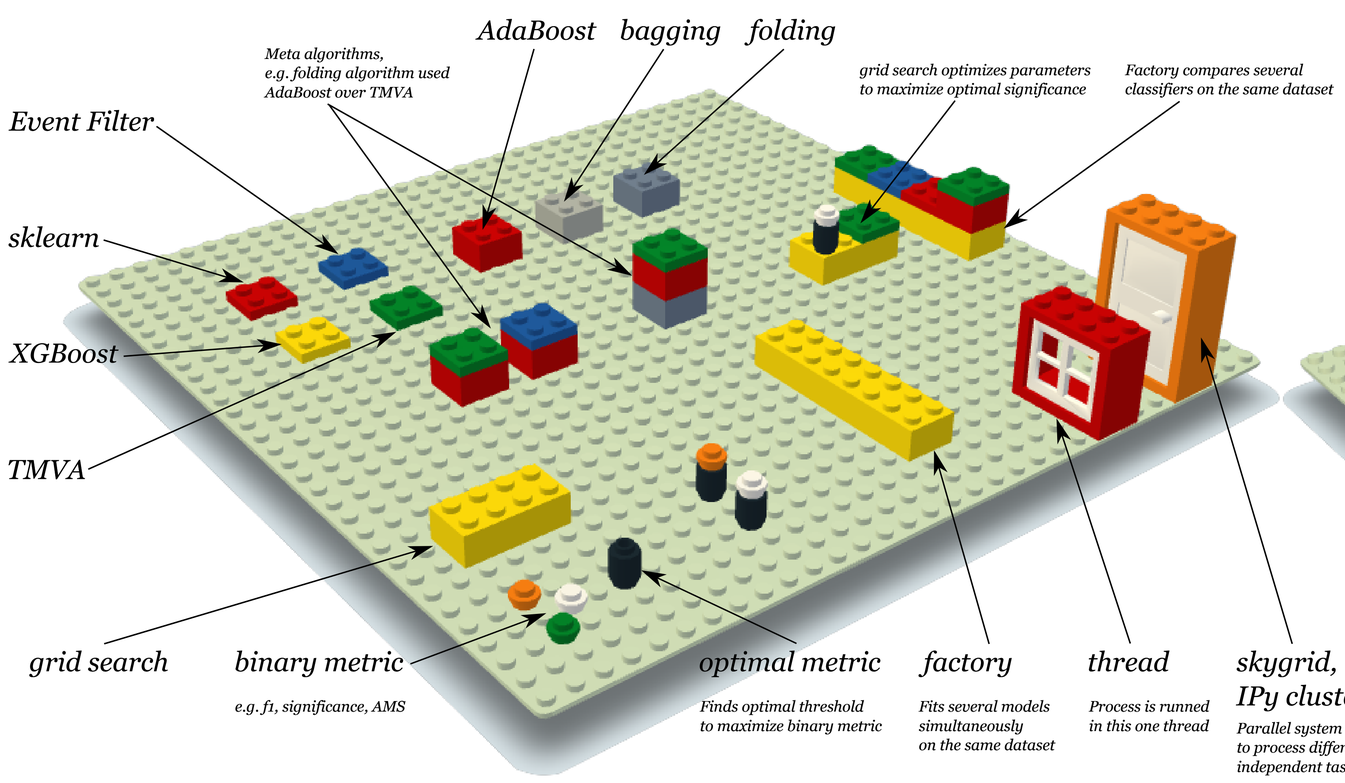
Yandex School of Data Analysis

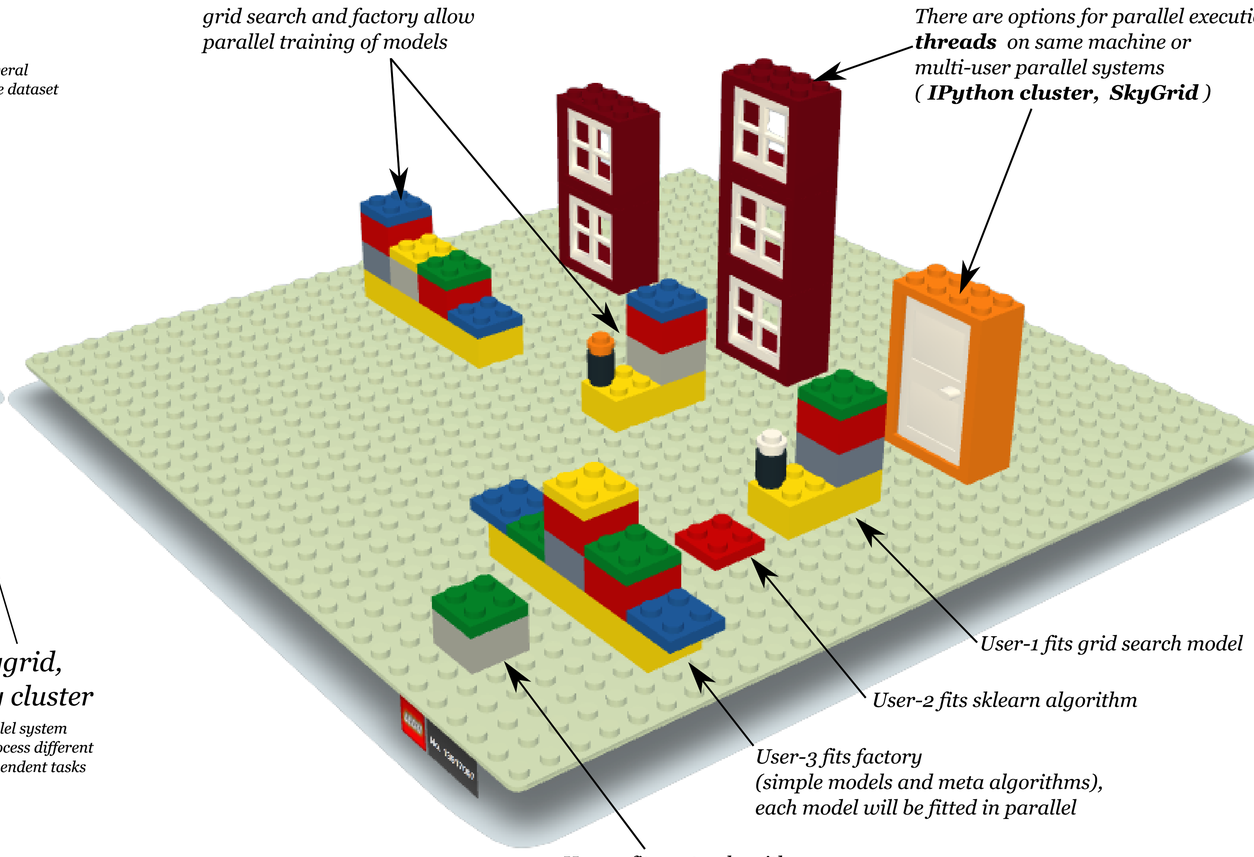
“Non-reproducible single occurrences are of no significance to science.” – Karl Popper

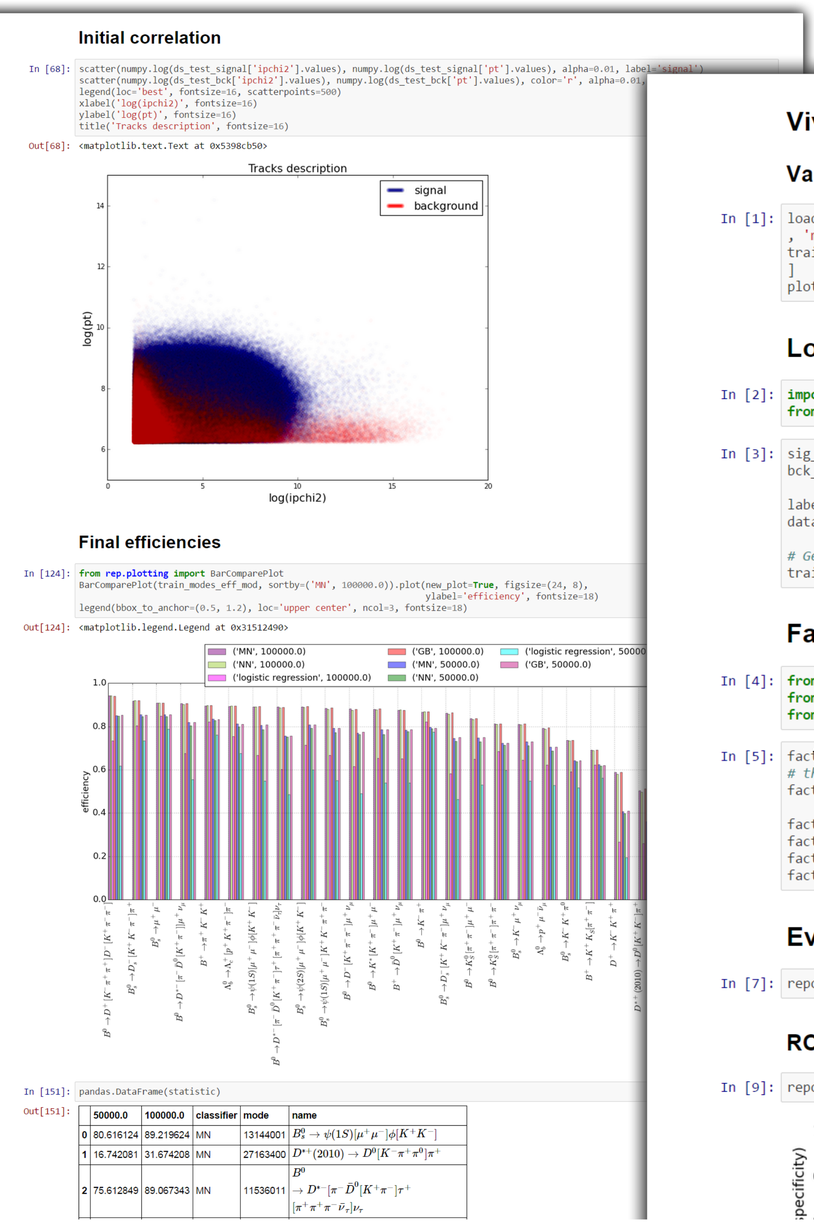
Plan for paper:

- Describe theory
- Some **magic** happens
- Conclusions









Initial correlation

```
In [68]: scatter(numpy.log(ds_test_signal['ipchi2'].values), numpy.log(ds_test_signal['pt'].values), alpha=0.01, label='signal')
         scatter(numpy.log(ds_test_bck['ipchi2'].values), numpy.log(ds_test_bck['pt'].values), color='r', alpha=0.01)
         legend(loc='best', fontsize=16, scatterpoints=500)
         xlabel('log(ipchi2)', fontsize=16)
         ylabel('log(pt)', fontsize=16)
         title('Tracks description', fontsize=16)
Out[68]: <matplotlib.text.Text at 0x5398cb50>
```

Tracks description

signal
background

log(pt)

log(ipchi2)

Final efficiencies

```
In [124]: from rep.plotting import BarComparePlot
          BarComparePlot(train_modes_eff_mod, sortby=('MN', 100000.0)).plot(new_plot=True, figsize=(24, 8),
                                                                            ylabel='efficiency', fontsize=18)
          legend(bbox_to_anchor=(0.5, 1.2), loc='upper center', ncol=3, fontsize=18)
Out[124]: <matplotlib.legend.Legend at 0x31512490>
```

('MN', 100000.0) ('GB', 100000.0) ('logistic regression', 50000
('NN', 100000.0) ('MN', 50000.0) ('GB', 50000.0)
('logistic regression', 100000.0) ('NN', 50000.0)

efficiency

```
In [151]: pandas.DataFrame(statistic)
Out[151]:
```

	50000.0	100000.0	classifier	mode	name
0	80.616124	89.219624	MN	13144001	$B_s^0 \to \psi(1S)[\mu^+\mu^-]\phi[K^+K^-]$
1	16.742081	31.674208	MN	27163400	$D^{*+}(2010) \to D^0[K^-\pi^+\pi^0]\pi^+$
2	75.612849	89.067343	MN	11536011	$B^0 \to D^{*-}[\pi^-\bar{D}^0[K^+\pi^-]\tau^+ [\pi^+\pi^+\pi^-\bar{\nu}_\tau]\nu_\tau$

Viva la Factory

Variables needed for analysis

```
In [1]: loaded_variables = ["FlightDistance", "FlightDistanceError", "IP", "VertexChi2", "pt", "p0
        , 'mass']
        train_variables = ["FL: FlightDistance/FlightDistanceError", "IP", "VertexChi2", "pt", "p0
        ]
        plot_variables = train_variables + ['mass']
```

Loading data

```
In [2]: import numpy, pandas
        from rep.utils import train_test_split

In [3]: sig_data = pandas.read_csv('toy_datasets/toyMC_sig_mass.csv', sep='\t', usecols=loaded_var
        bck_data = pandas.read_csv('toy_datasets/toyMC_bck_mass.csv', sep='\t', usecols=loaded_var

        labels = numpy.array([1] * len(sig_data) + [0] * len(bck_data))
        data = pandas.concat([sig_data, bck_data])

        # Get train and test data
        train_data, test_data, train_labels, test_labels = train_test_split(data, labels, train_si
```

Factory of different models

```
In [4]: from rep.metaml import ClassifiersFactory
        from rep.estimators import EventFilterClassifier, TMVAClassifier, SklearnClassifier, XGBoo
        from sklearn.ensemble import AdaBoostClassifier

In [5]: factory = ClassifiersFactory()
        # there are different ways to add classifiers to Factory:
        factory['ef'] = EventFilterClassifier(connection='test_connection',
                                              features=train_variables, iterations=200)
        factory.add_classifier('tmva', TMVAClassifier(NTrees=50, features=train_variables[:5], Shr
        factory.add_classifier('ada', AdaBoostClassifier(n_estimators=10))
        factory['xgb'] = XGBoostClassifier(features=train_variables[2:6])
        factory.fit(train_data, train_labels)
```

Everybody loves plots!

```
In [7]: report = factory.test_on(test_data, test_labels)
```

ROC curves (receiver operating characteristic)

```
In [9]: report.roc().plot(xlim=(0.5, 1))
```

Roc curves

specificity)

for name, cl in estimators.items():

Plot learning curves to see possible overfitting of trained classifier

Learning curves are powerful and simple tool to analyze the behaviour of your model.

```
In [36]: learning_curve = report.learning_curve(roc_auc, metric_label='roc auc', steps=1)
         learning_curve.plot(new_plot=True)
         Estimator tmva doesn't support stage predictions
```

Learning curves

ef
ada
xgb

roc auc

stage

Plot data information: features correlation matrix

```
In [21]: report.features_correlation_matrix_by_class(features=plot_variables).plot(new_plot=True, show_legend=False, figsize=(15, 5))
```

Correlation for bck events

Correlation for signal events

mass, dira, LifeTime, p2_pt, p1_pt, p0_pt, pt, VertexChi2, IP, FL

Plot distribution for each feature

```
In [23]: # use just common features for all classifiers
         report.features_pdf().plot()
```

REP was used for:

- Data popularity estimator github.com/hushchyn-mikhail/DataPopularity
- Bs –> μμμμ analysis
- Uboost-like algorithms github.com/anaderi/lhcb_trigger_ml
- LHCb topological trigger optimization

References:

- http://buildwithchrome.com
- http://jir.com